LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-678875

Statistical and Machine-Learning Classifier Framework to Improve Pulse Shape Discrimination System Design

R. Wurtz, A. Kaplan

October 30, 2015

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Statistical and Machine-Learning Classifier Framework to Improve Pulse Shape Discrimination System Design

Ron Wurtz and Alan Kaplan

Lawrence Livermore National Laboratory

Pulse shape discrimination (PSD) is a variety of statistical classifier. Fully-realized statistical classifiers rely on a comprehensive set of tools for designing, building, and implementing. PSD advances rely on improvements to the implemented algorithm. PSD advances can be improved by using conventional statistical classifier or machine learning methods. This paper provides the reader with a glossary of classifier-building elements and their functions in a fully-designed and operational classifier framework that can be used to discover opportunities for improving PSD classifier projects. This paper recommends reporting the PSD classifier's receiver operating characteristic (ROC) curve and its behavior at a gamma rejection rate (GRR) relevant for realistic applications.

Keywords: pulse shape discrimination, machine learning, classifiers

0 Introduction

The lead author has worked for years with some code that was written to do pulse shape discrimination (PSD), which is a variety of statistical classification, and was written by people unaware of the essential elements of classifier methods. Unlike a conventional classifier, these codes were unable to report, adjust, or compare performance in ordinary statistical terms. While designing replacement code built on ordinary statistical and machine learning principles, it made sense to write this limited primer on classifiers. The primer is aimed at people learning about ordinary classifier concepts for the first time, by way of PSD. It is intended to be comprehensive not in the sense of naming all the possible methods of machine learning (like neural nets, random forests, etc.) but the elements of a project that successfully implements a classifier (like feature selection, cross validation, and confusion matrices).

1 The PSD field

Pulse shape discrimination (PSD) is a variety of statistical classifier. PSD scintillators are sensitive to more than one kind of particle, and the time-evolution of the pulse of light created in the scintillator differs with different incoming particles (Wright 1956, Owen 1958, Brooks 1959, Horrocks 1973). The trick is to distinguish the different pulses of light in order to classify the different kinds of particles. Better scintillators make better-differentiated pulses, and there is always a hope that better capture and analysis of the pulses can improve the differentiation regardless of the scintillator. In the present era, much new work in PSD uses digitized pulses (*e.g*. Aleksan *et al*. 1988). Many tools have accumulated over the years to deal with digitized pulses and the classifiers that PSD researchers build. Because PSD system designers build statistical classifiers whether they know it or not, this paper suggests that the PSD field can easily adopt and adhere to statistical and machine learning principles.

Any new PSD classifier, or new variation on a PSD classifier, tends to start with the digitized pulse -- a time series of samples of a pulse -- where the sampling rate is fine enough to extract the crucial features of a pulse, notably the arrival time, the energy (size), and the particle type (shape). To "do PSD", a researcher will typically acquire a large set of such pulses under controlled conditions. The researcher will then separate the pulses into two groups – gammas and neutrons – using either a familiar method from the literature or by inventing a new method. Problems arise when a pulse's features fall in a region between the obvious gammas and the obvious neutrons. Also, digitizer vendors may provide tools that do a first pass to discriminate pulses (*e.g*. CAEN's DPP-PSD package, 2014) but the researcher cannot stop there: such vendor systems really only remove obvious unneeded events to reduce dataflow to a manageable level (much like trigger thresholding), and all collected pulses still require a complete classifier doing true shape discrimination. New PSD methods appear regularly in the literature. They fall into two general groups: common sense methods, and methods exploiting statistical and machine learning tools (see examples of PSD machine learning in Sanderson *et al.* 2012 and Yu *et al*. 2015). If the method is new, the researcher may write it up, describing how to use it and its capabilities. Sometimes, someone publishes a survey paper comparing several options for PSD, including the properties of the digitizer, or how the samples are combined for the discriminator, or the choice of discrimination (see Flaska *et al*. 2013 and Söderström *et al*. 2008).

Advances in PSD are weakened by the continual march of common sense disconnected from convention and rigor. Fortunately, statistical classification is a well-described field offering convention and rigor. Statistical classification encompasses not only advances in methods to perform classification, but also methods to plan for, carry out, and report the classification experiment regardless of advances. PSD can surely benefit by adopting a conventional approach to performing experiments to improve PSD classification. This paper describes the breakthrough of using statistical conventions to build a PSD classifier.

PSD already uses a few statistical conventions. These conventions are followed inconsistently in research papers. One good reason for the inconsistency is that these conventions often arose *ad hoc* for the PSD field and may be fairly weak. An excellent example is the figure of merit called "FOM", originally described in Winyard, *et al*. 1971, which is an easy-to-estimate separation of gamma distribution from neutron distribution, normalized by their spreads. Just a few years later, Sperr, *et al*. 1974 called the FOM "a rather specious quantity giving no meaningful indication of gamma-neutron discrimination". However, FOM persists, partly because the PSD field has yet to adopt conventional statistical methods. “Gamma rejection rate” (GRR) is another measure that is reported in one form or another in the literature by some researchers (Sperr *et al.* 1973, Mukhopadhyay *et al*. 2010, Kouzes *et al*. 2011). GRR is the rate of gammas misidentified as neutrons, per all gammas. Interestingly, GRR may be reported on the basis of a fit to a Gaussian for a distribution that has never been shown to be normal, and almost certainly is not: in such cases, a certain number of sigma cannot be related to a certain rate of misidentifications, and so sigma is misleading. This paper will go on to describe methods for reporting the performance of PSD that are similar to FOM and GRR and based on a firmer foundation.

There are two main kinds of systems that take a set of inputs and create an output based on training data. If the output is a continuous variable, the system is called a “regression” system. If the output is a set of symbolic classes, the system is called a classifier or classification system (if it covers only two classes, it is a “binary classifier”, if there are more than two classes, it is a “multiclass classifier”, or “M-ary classifier”). Just like the PSD systems stretching back into the 1950s, classifiers and regression systems are first designed, then trained, then run under normal operation. During normal operation, an electrical pulse is presented to the PSD system as a “sample” or “query” and the system “predicts” its particle type. In most early PSD system designs (see for example Gatti and De Martini 1962, and Owen 1962), the first step ran a regression system, which assigned a single value to the pulse based on a set of inputs, and then a binary classifier converted the estimated value to a class.

2 Classifiers

The analysis of any classifier problem and the design and implementation of its solution follow conventional procedures, even if the solution itself is a radical breakthrough. Classifiers are a subject of continuous study in both the fields of statistics and machine learning, and both fields use multiple terms to describe the same concepts. This paper will attempt to use and provide more universal terminology, though in reality, universal terminology does not exist. Numerous books, papers, and other materials (*e.g*. Kennedy and O’Hagan 2001, Ng 2014, Alpaydin 2004, Minka 2015) guide the perplexed through many classifier approaches, and those publications are written primarily for others attempting to move the classifier field forward. Such publications tend to downplay the complete tasks facing people about to build the classifier, such as the development of essential elements that support the classifier. (It is interesting to note that, because PSD researchers have vast sets of pulse data, PSD has gotten the attention of people interested in trying out new ways of crunching large data sets). In general outline, the tasks include (with examples drawn from PSD):

> Assess the design requirements (for example, the acceptable GRR) and the design constraints (for example, the maximum rate of events that the system can handle).

Acquire a large set of events under controlled conditions. These events may have been acquired in such a way that more information about each event is available than will be available under normal operating conditions. These events will be used to develop a quantitative description of the behavior of the system when presented with pulses.

Choose at least one classification algorithm matched to the characteristics of the data and the design constraints, with the prospect of meeting the design requirements.

Implement the algorithm and optimize its performance using the set of events and acting under the design constraints.

Assess the algorithm's optimized performance, and report on the performance and how well it met the design requirements. If this algorithm includes novel characteristics, its performance might be compared with other established algorithms.

Note that PSD can suffer if any these procedures, vague as they are, are ignored. In fact, there are a few essential elements for the development of a classifier, and employing them all leads to a useful outcome for the whole field. Alternatively, a partial report on painstaking work may be of little value to anyone until omitted portions are completed. As of 2015, the next PSD paper that reports on all of the essential elements will be the first PSD paper that reports on all of them.

3 Essential elements of classifier construction

This paper will now describe the essential elements in some detail. The elements are described roughly in order of the phases where each first arises. But first, an aside about the elements of a partial success. The most technically-exhilarating elements of the project are:

Developing derived features and/or labels (to satisfy constraints or to improve the ability of the measured features to describe the labels)

Selecting a model (or non-parametric model)

Defining the cost function (or its equivalent)

Selecting the optimizer (often determined by the choice of model and cost function)

A breakthrough with one of them will lead to partial success, and stopping there will lead to partial failure. Further, mistaking one element for another (like the model for the features) may attract partial failure. This paper is about preventing partial failures by stressing repeatedly the importance of choosing and building and reporting on all the phases that precede these elements (all the boldface entries above "model"), and to carefully perform and report on all the following phases that determine the value of the work (all the boldface entries below "optimizer"). A conscientious referee will cheerfully reject papers that leave out these essentials.

Design requirements. The design requirements are usually expressed in terms of a probability of identification or misidentification, and there is an excellent chance that, if there is more than one requirement, they are mutually exclusive. For PSD, a design requirement might be the highest acceptable rate of misidentified gammas per correctly identified neutrons. These kinds of practical requirements contrast with projects that are intended to determine if there is a more appropriate machine learning method to apply to a problem. In that case, the requirement is to predict what the optimal performance could be. Then the community can decide whether or not the performance is adequate to a design requirement.

Design constraints. The constraints of a design may drive the choice of algorithm and even drive the designer away from the optimum discrimination algorithm. Constraints have driven PSD design for a long time, particularly during the analog era. The most common constraints are space and time and data-rates, and limitations imposed by the hardware behavior. As an example, all the samples of all the pulses may overwhelm the readout, requiring compression (almost always lossy) just to move the data around. The compression itself may be optimized for the algorithm further down the line, but only if the constraint is known. Alternatively, the algorithm may be selected based on the knowledge that it is operating on compressed data.

Training sets. Training sets consist of the kind of information available about each pulse during the regular operation of the system for a large number of events. These data should well-represent the data that the PSD system will be operating on. The training pulses should extend over the full energy range expected, and if a significant amount of pile-ups, saturated events, MIPs, etc., are expected to be classified, the training dataset should include them. Training sets are also, in most cases, augmented with information that is not collected during regular operation for various reasons. The reasons include constraints during regular operation that reduced the amount of information available per event. Training sets may also include information that is extracted by means other than the regular operation of the system. These datasets are easily processed when formed into a two-dimensional matrix where each row is an event and the columns are the pieces of information about the event. This matrix is sometimes called a "**design matrix**". The rows of a design matrix, which for digital PSD are a vector of samples of each pulse event, are sometimes called "**examples**" or "instances", and the columns are called "features".

Incomplete data. One element of classifiers that is rare if not nonexistent in PSD applications is a method for dealing with a training set containing incomplete data. This problem arises for example in time-series applications. We include this element only for completeness.

Cleaning. It is common to "clean out" events preemptively during the regular operation and during the collection of the test sets. Essentially this means that the builder already has a crude but effective binary classifier describing events of interest or of no interest. When using a digitizer, this is as simple as setting the threshold to retain only the pulses of a certain height, where dealing with the smaller pulses presents problems of high data-handling rate and noisy impossible-to-classify pulses. Some digitizers offer a method of flagging pileup pulses, and the user can decide whether to reject or retain these pulses. It is possible that after collecting events, the features of certain events will cause them to be flagged as too hard to deal with, and so the builder will choose to remove them from the training set, which means that events like them, collected during regular operation, will be beyond the bounds of the classifier as well, The issue of out-of-bounds data is called **coverage**.

Features. The raw data read off the sensor are "**features**". In addition to the samples of the pulse, a digitizer system may report a pile-up flag, peak location, saturation, etc. Not all the features measured by the system need to become part of the design matrix. If for some reason a feature is deemed insignificant, it may be best to remove it, especially if operating under a design constraint. Methods for discovering significant features are covered by feature learning or feature engineering.

Transformed feature. If a column in the design matrix is derived from the raw features, it too is another feature, which this paper will call a "transformed feature". Transformed features are commonly used to reduce dimensionality and maximize linear independence of the raw features. Sometimes a feature is transformed non-linearly into another feature so that classifier can be dealt with as a linear function. Another use for derived features is to compress some of the features into a smaller footprint in order to meet a design constraint. Machine learning often deals with improving features by reducing dimensionality, making linear functions, increasing orthogonality, etc. to make the problem "easier". A somewhat outdated (but still used) technique for the transformation is "feature engineering", where the human chooses a transformation that essentially computes a relatively small number of features. This approach can be viewed as: "let's find a small number of quantities that enable efficient classification." For example, in the case of pulse features, it is common to compute the bias from a pre-trigger and remove the bias from the rest of the samples of the pulse, and, further, compute a "tail-to-total" derived feature that is easily converted to particle type. The problem with this approach is twofold. First, you cannot gain information with any transformation of the data (due to the data processing inequality), so you may be limiting performance right off the bat. Secondly, those features that a human may think are important to the problem may not be. The more modern approach is to use lossless transforms that make the data more amenable for classification. Examples of these include Fourier transforms and wavelets. Modern classification methods are capable of dealing with the complexities and size of real data. PSD papers describing a method for using principal component analysis (PCA), or a method for extracting wavelets (see

Yousefi and Lucchese 2009), are actually describing a way of obtaining transformed features with the hope of improving compression or improving convergence. Another variation of transformed features are the outcome of a “generative neural net”, where the next to last layer is used as the input features to the classifier system.

Label. The information that is to be extracted from the features is called an “**output**” or a “**label**”. Labels are not collected during regular operation (otherwise, the classifications would be measured directly and no one would need to build the classifier). PSD labels are at least “neutron” versus “not neutron”. In the case of only two possible labels, the classifier is a “**binary classifier**”. If the builder defines other possible labels (like “pileup” or “MIP” or “saturated”), then the classifier is called a “**multiclass classifier**”. The approach of training classifiers using data containing ground truth labels is called **supervised learning**. It is important to be aware that any method of labeling may result in a few mis-labels that will affect the training of the system (see Sanderson *et al*. 2012 for a means of estimating mis-labeling). In PSD, sometimes the events are labeled by time-of-flight methods that are unavailable during regular operation (time-of-flight labeling has a very low yield, meaning that to get millions of labeled events, one may need to collect many billions of events). Sometimes PSD events are labeled by the method of looking for clusters (see Luo *et al*. 2010). In the case where training labels are unavailable, techniques falling under the name of **unsupervised learning** may be used. The goal of unsupervised learning is to discover groupings, or clusters, within the training data without access to ground truth labels. These groups may then be linked to meaningful labels, or used directly as outputs. It can be difficult, however, to access the performance of an unsupervised learning method in an unbiased manner, since here ground truth does not exist.

Validation and test sets. Training sets are further separated into two or three subsets for test purposes. Testing on the training set would inevitably lead to exaggerated performance, since the classifier would have the advantage of seeing the data beforehand (it is possible to build classifiers that perform arbitrarily well when testing on the training set). Recall that the training set is used to demonstrate to the algorithm the connection between the features and the labels. In order to test the trained algorithm, the **test set** is used to check to see at what rate the correct label is recovered. In the event that a model will be evaluated compared to another model (“model selection”), the builder needs a cross validation set (or just “**validation set**”) to compete the two models. In the event that the algorithm will be evaluated on its own, the builder needs only a test set.

Classifier family. The classifier itself is an algorithm that takes a vector of features and outputs an estimated label. Any classifier may be specified by a decision boundary in the feature space. In the case of binary classification, this would assign one of two outputs to every possible input (feature vector). In the training phase, the training data are used to select a classifier, with to goal of maximizing performance on new test data (not the same training set). One widely used approach is to initially select a family of classifiers, such as decision trees, neural networks, or support vector machines. These families are specified by a set of parameters. The training phase then selects one classifier by estimating optimal values for the parameters. It is often difficult to know beforehand which family to use. Here, intuition plays a large role. Each of the families mentioned constructs boundaries that look different. For example, decision trees can construct rectangular boundaries, whereas neural networks can construct boundaries made up of intersecting lines (polygons). Combinations of these families can also be considered. Another option rooted in the theory of pattern recognition is to consider families generated by probabilistic models of the data. In this approach, decision boundaries are constructed using likelihood ratios between probability distributions for each class, which are estimated from the data.

A basic result in the theory of machine learning is that the optimal decision boundary (the one that minimizes the probability of classification error) is generated by selecting, for a given vector of features, the class maximizing the posterior probability over all classes (*e.g*. Devroye *et al*. 2013). Realization of the optimal classifier, however, requires knowledge of the underlying probability distributions, or models, for each class. Motivated by this result, one approach to designing classifiers is to estimate a parametric or non-parametric model for each class from training data, and use the estimated models in a “plug-in” decision rule. Two sources of error can be identified in this regime: systematic errors and estimation errors. Systematic errors arise from deviations in the form of the chosen models compared to the true models, while estimation errors capture the deviations between the estimated models and the best possible models within the chosen form (this is also known as the bias *vs* variance tradeoff in statistics). Estimation errors can be reduced by increasing the amount of training data. These errors can then be mapped onto the resulting classification error.

Models. Classification systems can be based on a model or not, and a model can be **parametric** or **non-parametric** (*e.g*. Fukunaga *et al*. 2013). A model is formally specified by a probability distribution over all possible data points (samples). In some cases, parametric models may be derived from underlying physical laws, and in other cases, they may be purely phenomenological, like templates. In either case, such a model summarizes the training set more compactly. Once the form of the model is chosen, its **parameters** are estimated from training data (see, for example, The n_TOF Collaboration, *et al*. 2002). The model with estimated parameters is sometimes called a “hypothesis function” and the value the hypothesis function returns for a vector of features is called a “prediction”. One of the central issues surrounding model selection and fitting is that as the model complexity (number of parameters) increases, the model fit almost always also increases. Techniques from model selection (such as the Bayesian Information Criterion) may be used to access an appropriate level of complexity necessary to adequately capture variations in the data. Using a model that is too complex may lead to overtraining, causing a degradation in classification performance. Classifier systems that do not use a model at all include neural nets and support vector machines.

Optimization. Estimating parameters in the modeling process ultimately reduces to an optimization problem to meet an objective. (The **objective function** goes by many names, including "cost", "utility", "loss", and "payoff" function). The problem is to maximize the likelihood of the data given the model over parameter values. In some cases, the resulting optimization problem takes familiar forms, such as a minimization of a sum of squares ("chi-squared" objective). For example, this occurs when the data are modeled as a Gaussian random vector with independent components. Under more complex models (such as mixture models, for example), the objective function becomes more complex or even intractable for analytic computation. If this is the case, then sampling techniques exist to arrive at approximate solutions. Variational methods may also be used to approximate the objective function by a tractable one (*e.g*. Jordan *et al*. 1999). For classifiers that do not estimate models for each class, such as **neural networks** and **support vector machines**, the relevant optimization is called empirical risk minimization, in which classification error over the testing set is directly minimized. The tradeoff between classifier (or classification boundary) complexity and bias in overtraining is described by Vapnik-Chervonenkis (VC) Theory (*e.g*. Vapnik *et al*. 1998). An optimizer is thought of as very good if it both converges quickly and actually finds the absolute minimum. If one is building a classifier for PSD, the optimizer's speed of convergence may be of minor importance unless a classifier needs to be re-trained on a timescale shorter than the convergence time.

Threshold. For binary classification problems, optimality may be specified in terms of detection rate for a fixed false alarm rate (this may also be performed for an M-ary classification problem by posing it as a sequence of binary problems). In this case, the optimal form for the classifier is constructed by comparing the log-likelihood ratio between the two classes to a threshold (this is known as the Neyman-Person lemma). This type of analysis leads to ROC curves (see "trade curves" below). A threshold of zero leads to the minimum error probability classifier. Adjusting the threshold effectively moves the decision boundary and creating a new detection rate *vs* false alarm rate operating point. (When the algorithm uses a model, the objective function's parameters include both the model's parameters and other weighting parameters that describe the cost of misclassification, which include "regularization").

Training phase. As noted above, depending on the experiment, there are two or three phases that use training data. The training phase, sometimes called "learning" is the optimization phase.

Validation phase. The validation phase is optional and occurs when there is more than one model to compare the performance of.

Test phase. The test phase is the process by which the tuned parameters are converted into performance measures, which can be converted into numbers that can be compared with design requirements. For an already-built system, this phase can stand alone to provide predictions for a test set to tune operation parameters and report performance.

Confusion matrix. The confusion matrix is a venerable statistical tool describing how well the classifier system was able to match the test set to its labels. The confusion matrix is where the project returns to its source to see if it is capable of meeting the design requirements. For a binary classifier, the custom is to define the two labels as "positive" and "negative", and then arrange the outcomes from the test set in a two-by-two grid. The correctly-identified examples are called "true" and the mis-identifications are called "false", as shown in the following diagram.

True positives **TP**	**False positives** **FP**
False negatives **FN**	**True negatives** **TN**

These test measurements can be used to derive experimentally-determined rates of identification and misidentification. There are a few commonly-quoted rates. This method of reporting the performance of classifiers is so widespread that different fields have given the same rates different names. The rate of true positives per all positives (TP/(TP+FN)) is called the "true positive rate" (TPR), the "detection rate" (P_{det}), and the "recall", just to name three. If neutrons are "positives" and gammas are "negatives" (and there are no other options) then PSD's GRR is FP/(FP+TN), also called the "false positive rate" (FPR) which is also 1 minus the "specificity". In the event that there are N>2 classes, the grid grows to N-by-N, and performance is assessed by conventional multiclass classifier means (e.g. "one versus rest" or "one versus one"). It is important to note that the confusion matrix can change drastically when the decision boundary moves, meaning that a classifier system does not produce a single confusion matrix, but a family of matrices that is better characterized by the trade curves.

Trade curves. It should be fairly obvious that one can set a misclassification parameter (commonly represented by the "threshold" in PSD) in such a way that GRR seems to be reduced to zero. In the real world, when GRR or FPR goes not to a small number but to zero, the boundary between the positives and the negatives has moved to a place where no sample is classified as a "positive", causing TPR also to drop to zero. If TPR is required to be greater than zero (almost a certainty) then the trade between FPR and TPR is a "multi-objective optimization" problem, and the trade behavior is summarized in a curve of FPR versus TPR parameterized by the location of the boundary. That specific curve is called a "ROC curve", and it represents a small family of informational curves called "trade curves". These curves describe the trades that occur in the confusion matrix as misclassification parameters are swept from one extreme to the other (note how Sanderson *et al*. builds a ROC by varying the ratio of C_+ to C_-). Another very common trade curve is the precision-recall curve, or "P-R curve". The ROC curve depends only on the internal rates within the distributions of the positives and negatives. The P-R curve, because "precision" is defined as TP/(TP+FP), depends on knowing the relative rates of positives and negatives, since TP are actually positives and FP are actually negatives. If the expected normal operation rates are different from the rates in the test set, the numbers TP+FN (*i.e.* all neutrons) and FP+TN (*i.e.* all gammas) must be scaled to the expected positive and negative rates. Such scaling changes the P-R curve. The trouble with precision is that it is highly sensitive to the ratio of positives to negative in the sample data. In many cases, the number of negative samples is much higher than positive samples. This imbalance is heavily abused in the literature and requires careful treatment.

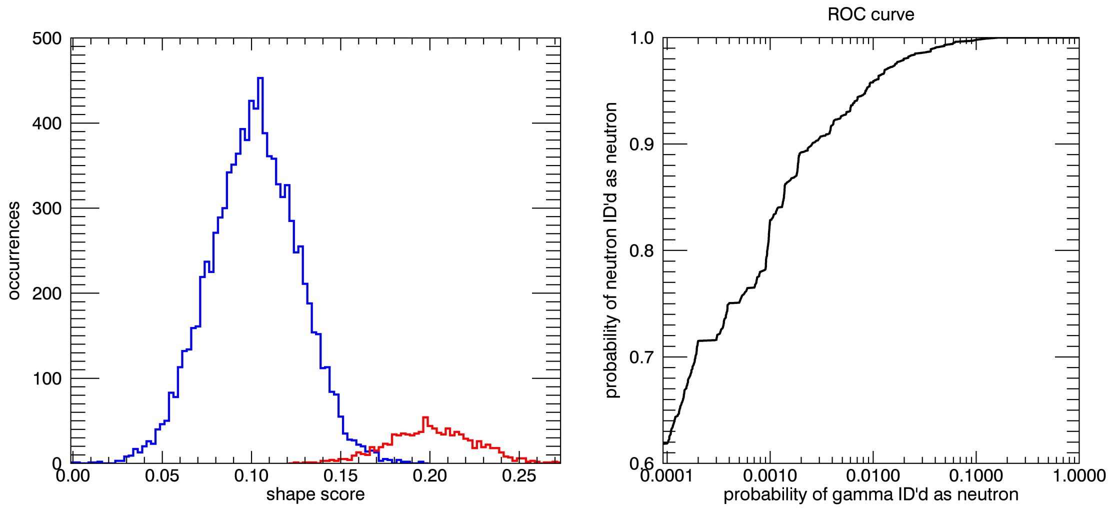

Figure 1: On the left are simulated distributions of gammas' pulse shape scores (in the taller peak) and the neutrons' pulse shape scores (in the shorter peak). The ROC curve on the right is derived from the distributions on the left, and shows a case where it is hard to balance ideal mis-identification rates. In this simulation, 40% of neutrons are lost if the gamma rejection rate is 1:10,000.

Figure of merit. PSD already has a figure of merit. In the ideal case that the distributions of gammas and neutrons are normal distributions, the FOM is a reasonable summary of performance for comparing among fundamental design choices. However, in the real world, it is preferable to remove the assumption that the distributions are normal and instead report the ROC curve. To reduce the performance to a single figure, the ROC curve has a summary statistic called the AUROC, for "area under ROC", that is similar to the FOM (because AUROC increases for better separation of the distributions) and not bound to normal distributions. The P-R curve has a statistic known as "F_1" that is maximum at a combined best precision and recall (depending on the evaluator's needs, the F-statistic can be weighted toward P or R). Another statistic is the "equal error rate" (EER), the value of P_{FA} where $P_{FA} = 1 - P_{det}$. Often the chosen figure of merit depends on the culture of the community in which results are published. The AUROC, maximum F_1, and EER – and for that matter, the FOM – fail to describe the low-GRR regime, because they describe a capability in regimes where the costs of gamma and neutron misidentification are roughly equal. For practical PSD purposes, systems sensing fast neutrons often require an unequal trade between TPR and FPR that results in extremely low GRR and hopefully not very low TPR. To summarize, a single figure of merit has trouble representing performance except relative to another combination of scintillator, digitizer, and classifier algorithm (*e.g*. Pawełczak *et al*. 2013).

This paper recommends that future published PSD reports always include the ROC curve, because of the challenge of multi-objective optimization, along with the FPR (GRR) *and the TPR* at a representative low-GRR threshold, say GRR=1e-5. (1e-5 seems natural when particles are dominated by gammas and the total particle detection rate is near 1e6/sec – that leads to approximately one misidentification per second). Further, since the very low GRR regime is often the most interesting, the ROC curve should be plotted logarithmically in GRR. (For work whose design requirements are sensitive to populations of mis-identified gammas among the neutrons, it may also be helpful to present a P-R curve.) Be aware that claims of one-in-a-million for GRR require a few million examples in the test set, and claims of GRR = 0 are impossible to make, and instead a bound must be stated, *i.e.* GRR < 1/(FP+TN). A further complication arises at low signal-to-noise: the performance degrades as a function of energy. See the "energy dependence" discussion below.

Prediction. Finally the entire project is put into action in **normal operation**. As each new pulse (sometimes called a "**query**" or "test instance" or "sample") arrives, its raw features are converted to the features used by the classification process, and the model and its optimized parameters operate on the features to perform a "prediction" to classify the pulse.

4 Specific challenges for PSD classifiers

4.1 Energy dependence

There are a few PSD-specific challenges relevant to classifiers. The first issue has to do with the increase in signal-to-noise in the digitized pulse with increasing energy, meaning that high energy pulses have distinct shapes by particle types, while the particle types of low energy pulses become less distinct with decreasing energy. It is common to solve this problem by creating subregions by energy, and training separate classifiers for each subregion. It is obvious that each subregion will have a different trade curve, and so if one value of TPR is selected as a condition of operation for the whole PSD system, then each energy subregion will have a different FPR. It is possible to construct a single classifier that works over all energy regions. It would "know" that lower energy regions are harder to separate. One approach is to use a bank of classifiers by designing of a set of differently-labeled subregions, or by using a decision tree.

4.2 Multiple detectors in one system

The second issue deals with a dilemma similar to the energy subregion problem. When using several scintillator/PMT/digitizer assemblies, each assembly will have slightly different performance. Here again the ROC curves will be different and the operator will have to make a decision about the tradeoffs to satisfy design requirements.

4.3 Pile-ups

Pile-ups lead to yet another similar problem. When the system is in the presence of a high rate of particles (typically gamma rays), two or more consecutive overlapping pulses, called "pile-ups", become a significant classification issue (note that Kaplan *et al*. 2013 started looking at this problem). Many of the lowest-energy gamma rays do not trigger the system, and are even incapable of triggering the digitizer's pile-up flag because they do not represent a tall-enough or steep-enough edge similar to the normal trigger. However, an undetected pile-up or two following a gamma creates an excess of light in the later portion of the pulse, making it similar to a neutron pulse with an excess of delayed light. For the classifier, this means that the rate of borderline pulses goes up, and the misidentification rates represented by the confusion matrix are different for different pile-up conditions. Making the problem harder, it may be impossible to perform supervised labeling of unflagged pile-ups because if we could have flagged them, we would have. If the analysis relies on a predicted rate of false neutrons, the rate has to be predictable. The simplest solution is to obtain the training data under the same rate of pile-up as the data during regular operation, though that solution is seldom easy in practice.

4.4 Comparison across systems

Finally, PSD has always suffered from the problem of comparison across systems. Anyone wanting to report on the performance of a new component must insert it into a combination of scintillator / electronics / algorithm, where it will be difficult to isolate that component's contribution to the figure of merit, and impossible to compare it with another researcher's published performance. A new FOM may be proposed (*e.g.* Söderström *et al*. 2008 or Flaska *et al*. 2013) to bring out one or another characteristic of performance – or even be incorporated into a cost function – and yet a new FOM will probably be monotonic with the established FOM, and still does not solve the problem of universal comparison. We offer no solution to this problem (though, if pressed, this paper would say to cite TPR at GRR=1e-5), however, it is important to acknowledge that there is a problem of scoring in PSD that might be solved by talking things over with a statistician or signal processor.

5 Conclusion

PSD is an ideal field for developing robust new classifiers using convention and rigor, and PSD researchers should explore and exploit the wealth of classifier methods whenever they can. This paper presented a primer of the elements of a fully-realized project to design, build, implement, and report performance for a classifier, with examples from PSD. This paper also recommended reporting the PSD classifier's ROC curve and its behavior at low GRR, in order to define the performance of any proposed new PSD method. The authors remind the reader that each of the elements in this brief paper represents several huge discussions, many of them taking up entire semesters – the paper raises awareness of all the elements, and cannot be relied on to explain how to carry them out. A high-level statistical software package may provide a useful entry to learn about the options for the various elements and to prototype a complete classifier system as laid out here. The authors expect that the reader immediately and enthusiastically runs to the computer or lab and rebuilds a PSD project along conventional classifier lines to produce a thing of beauty.

Acknowledgements

Over the years, RW had numerous discussions with Karl Nelson, Jim Candy, Vera Bulaevskaya, Mike Zelinski, and Kristin Lennox who carefully explained very simple concepts repeatedly until he partially understood them. RW also benefited from talking about Jason Newby and Jerome Verbeke's PSD algorithm with Serge Ouedraogo, from Nathaniel Bowden regularly giving him difficult-to-parse PSD papers to read, from Andrew Glenn's requests to make the most of digitizer compression, and from requests by Les Nakae and later Gary Guethlein to provide conventional statistics describing the performance of a PSD system. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

References

R. Aleksan, J. Bouchez, M. Boussicut, T. Desanlis, D. Jourde, J. Mullié, F. Pierre, L. Poinsignon, R. Praca, G. Roussel, J.F. Thomas, "Pulse shape discrimination with a 100 MHz flash ADC system", 1988, Nuclear Instruments and Methods in Physics Research A:, Volume 273, Issue 1, 1 December 1988, Pages 303–309

E. Alpaydin, 2004. "Introduction to Machine Learning" MIT Press, 2004.

F. D. Brooks, "A scintillation counter with neutron and gamma-ray discriminators," Nuclear Instruments and Methods, vol. 4, no 3, Apr. 1959, pp. 151-163.

CAEN corp. DPP-PSD Digital Pulse Processing for the Pulse Shape Discrimination. Available: http://www.caen.it/csite/CaenProd.jsp?parent=39&idmod=770 retrieved 2014.

L. Devroye, L. Györfi, G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Vol. 31. Springer Science & Business Media, 2013.

M. Flaska, M. Faisal, D. D. Wentzloff, S. A. Pozzi "Influence of sampling properties of fast-waveform digitizers on neutron–gamma-ray, pulse-shape discrimination for organic scintillation detectors", Nuclear Instruments and Methods in Physics Research A:, Volume 729, 21 November 2013, Pages 456–462

K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic press, 2013.

E. Gatti and F. De Martini, "A new linear method of discrimination between elementary particles in scintillation counters, Nuclear Electronics", vol .2, IAEA Vienna, 1962. pp 265-276.

D. L. Horrocks, "Applied Liquid Scintillation Counting," in Liquid Scintillation Counting, volume 3, Crook, M. A. and Johnson, P. eds. Heyden, London. pp 3-20, 1974.

M. I. Jordan et al. "An Introduction to Variational Methods for Graphical Models." *Machine learning* 37.2 (1999): 183-233.

A.C. Kaplan, M. Flaska, A. Enqvist, J.L. Dolan, S.A. Pozzi, “EJ-309 pulse shape discrimination performance with a high gamma-ray-to-neutron ratio and low threshold”, Nuclear Instruments and Methods in Physics Research A, Volume 729, 21 November 2013, pp 463-468

M. C. Kennedy, A. O’Hagan, “Bayesian Calibration of Computer Models” 2001. J. R. Statist. Soc. B. 63, part 3, pp 425-464.

R. T. Kouzes, J.H. Ely, A. T. Lintereur, E. K. Mace, D.L. Stephens, M. L. Woodring, “Neutron detection gamma ray sensitivity criteria”, Nuclear Instruments and Methods in Physics Research A, Volume 654 (2011) pp 412-416

X. Luo, G. Liu, J. Yang “Neutron/Gamma Discrimination Utilizing Fuzzy C-Means Clustering of the Signal from the Liquid Scintillator” 2010 First International Conference on Pervasive Computing, Signal Processing and Applications, IEEE

T. Minka, “A Statistical Learning/Pattern Recognition Glossary”. Retrieved 2015. http://alumni.media.mit.edu/~tpminka/statlearn/glossary/

S. Mukhopadhyay, J. Glodo, R. Hawrami, U. Shirwadkar, E. van Loef, W.M. Higgins, A. V. Churilov, K.S. Shah, “Detection of Nuclear Material with Dual Neutron-Gamma Detector”, IEEE International Conference for Technologies on Homeland Security, 2010, pp 404-409

The n_TOF Collaboration, S. Marrone, D. Cano-Ott. N. Colonna, C. Domingo, F. Gramegna, et al, “Pulse shape analysis of liquid scintillators for neutron studies,” Nuclear Instruments and Methods A, vol. 490, Sept. 2002, pp. 299-307

A. Ng, “Machine Learning Course”. Coursera Online Courses, retrieved 2014, https://class.coursera.org/learn/machine-learning

R.B. Owen, “Decay Times of Organic Scintillators”, IRE Transactions on Nuclear Science, 1958, NS-5, 198-201

R. B. Owen, “Pulse Shape Discrimination – a Survey of Current Techniques” IRE Transactions on Nuclear Science, Volume 9, Issue 3, 285 – 293 June 1962

I. A. Pawełczak, S. A. Ouerdraogo, A. M. Glenn, R. E. Wurtz and L. F. Nakae, “Studies of neutron-gamma pulse shape discrimination in EJ-309 liquid scintillator using charge integration method,” Nucl. Instr. Meth. Phys. Res. A 711, 21, 2013.

T. S. Sanderson, C. D. Scott, M. Flaska, J. K. Polack, and S. A. Pozzi , “Machine Learning for Digital Pulse Shape Discrimination,” Proc. IEEE Nuc. Sci. Symp., p 199-202, 2012.

P.-A. Söderström, J. Nyberg, R. Wolters, "Digital pulse-shape discrimination of fast neutrons and gamma rays," Nucl. Instr. Meth. A, 594, 79-89, 2008.

P. Sperr, H. Spieler, M.R. Maier, D. Evers, "A simple pulse-shape discrimination circuit", Nuclear Instruments and Methods, Volume 116, Issue 1, 15 March 1974, Pages 55-59

V. N. Vapnik, V. Vapnik. *Statistical Learning Theory*. Vol. 1. New York: Wiley, 1998.

R.A. Winyard, J.E. Lutkin, B.W. McBeth, "Pulse Shape Discrimination in Inorganic and Organic Scintillators. I", Nucl. Instr. and Meth., 95 (1971), p. 141

G.T. Wright, "Scintillation Decay Times of Organic Crystals" Proc. Phys. Soc. B 1956, 69, 3, 358-372.

S. Yousefi, L. Lucchese, "Digital discrimination of neutrons and gamma rays in liquid scintillators using wavelets," Nuclear Instruments and Methods A, vol.598, no. 2, Jan. 2009, pp. 551-555.

X. Yu, J. Zhu, S. Lin, L. Wang, H. Xing, C. Zhang, Y. Xia, S. Liu, Q. Yue, W. Wei, Q. Du, C. Tang "Neutron‑gamma discrimination based on the support vector machine method", Nuclear Instruments and Methods in Physics Research A:, Volume 777, 21 March 2015, Pages 80‑84.